Brett Sherman's Rock and Roll Guide to Soundness and Completeness

So we gave you all these rules that allow you to write a proof of some conclusion from some premises. And we told you that whenever you can write such a proof, the conclusion is a consequence of the premises (*soundness*). And we told you that whenever some sentence is a consequence of some other sentences, you can write a proof from the premises to the conclusion (*completeness*).

Now, it would be okay if you simply took our word for it—we weren't lying. But, it's important to understand just how substantial our claims are. And given how substantial they are, it's important to see how you could prove such claims.

Why soundness and completeness aren't trivial and obvious

Let's start with soundness. Our system of rules is a lot like that word game where you're given a word and you have to get some other word by changing one letter at a time, such that after each change you still have a proper word. For example, you can derive DOGS from CATS as follows:

CATS
COTS
COGS
DOGS

In this game, you start with certain symbols and you have rules for manipulating those symbols. But the moves that you're allowed to make don't tell you anything interesting about the meanings of the words you're playing with. You don't learn anything about the relation between dogs and cats simply from the fact that you can derive DOGS from CATS.

Our system of rules also allows us to manipulate symbols. However, unlike in the above game, we want the manipulations to tell us something about the meanings of the symbols involved. Specifically, we want the manipulations to tell us which sentences are consequences of which sentences. But in order for our manipulations to tell us this, we need to be really careful about which rules we use. We can only use those rules that will take us from some sentences to a consequence of those sentences. And that's not a trivial task.

To see why, consider the following bad rule:

BAD CP: If on some line (m) you assume ϕ, and on some line (n) you derive ψ, then on a new line (o) you can write $\phi \rightarrow \psi$, where the dependency numbers of (o) are the dependency numbers of ϕ minus the dependency numbers of ψ.

This rule is not obviously bad. Even if you're able to spot the mistake now, you probably wouldn't be so quick in spotting the mistake if we hadn't given you a good CP rule. The following proof invoking BAD CP should make clear why this rule doesn't do what we want:

1.	(1)	P	A
2.	(2)	~P	A
2.	(3)	~P → P	1,2 BAD CP

According to line (3), ~P → P is a consequence of ~P. But that's wrong. Whenever ~P is true, ~P → P is false. The flaw in BAD CP lies in its recipe for calculating the dependency numbers. The bad rule tells you to take the dependency numbers of the antecedent minus those of the consequent. But we all know that it should be the other way around.

Now, it's easy to see that a particular rule is flawed once such a problem is pointed out. But a more general worry persists: how do we know that we've removed all of the flaws in all of our rules? How do we know that we don't need to add some additional restriction on the usage of some rule? Our proof of soundness is aimed at removing this worry.

Completeness is perhaps even more exciting.

Suppose that we gave you all of the rules that you now have except for UI. The system would be sound—every permissible move that you make will take you from some sentences to a consequence of those sentences. But there would be lots of valid arguments that you wouldn't be able to prove. For example, the following argument would not be provable (try proving it without using UI):

(1)	(x)(Fx → Gx)		
(2)	(x)(Gx → Hx)	//	(x)(Fx → Hx)

But the conclusion is obviously a consequence of the premises. If all monkeys are swimmers, and all swimmers eat bananas, then it follows that all monkeys eat bananas. While we're coming up with a bunch of rules, we might as well add enough rules to allow us to prove the above argument.

Now, we can add UI to our system, allowing us to prove the above argument. But once again, a more general worry persists: how do we know that there aren't other valid arguments out there that we can't prove? How do we know that we don't need to add any additional rules to our system? Our proof of completeness is aimed at removing this worry.

Proving soundness

So we want to show that if there is a correctly written proof from premises A_1, …, A_n to conclusion B, then B is a consequence of A_1, …, A_n.

Here's another way of putting what we need to prove. Take any line at all of some proof. It might look like this:

1,4,7	(9)	P	2,5 MPP

Let's introduce the following shorthand: let **Q[n]** denote the conjunction of the sentences that line (n) depends on. So in our above example, Q[9] stands for the conjunction of the sentences on lines (1), (4), and (7).

Now, on every line of every proof, the sentence on the line should be a consequence of the sentences that it depends on. So, in other words, for every line (n), Q[n] should imply whatever sentence is written on (n). If so, let's say that line (n) is a *good line*.

What we want to show in order to prove soundness is that *if you start with one or two good lines, and you apply one of our rules, the line you get as a result is also a good line.* If that's the case, then whenever you have a correctly written proof, the conclusion will be a consequence of the premises.

Now all we do is take each rule and prove that if you start with one or two good lines and apply it, the result is a good line. Different rules will require different proofs, but a couple of examples will make clear what the proofs look like.

Rule of Assumptions

This rule allows us to write anything we want at any point in a proof, so long as the line we write looks like this:

n	(n)	Φ	A

In order to show that the rule is sound, we need to show that the line that results from using the rule is good. And it should be pretty obvious that line (n) is a good line. Remember, in order to show that a line is good, we need to show that Q[n] implies the sentence on line (n). Since Q[n] = n, we need to show that Φ implies Φ. And it does. Whenever Φ is true, Φ is true. So the Rule of Assumptions is sound.

&-Elimination

So we're given line (m), and we use the rule to get line (n). We assume that line (m) is good. We need to prove that line (n) is good.

X	(m)	$\Phi \& \Psi$	
X	(n)	Φ	m, &E

First note that Q[m] = Q[n]. Since by hypothesis, line (m) is good, we know that Q[m] implies $\Phi \& \Psi$. Since Q[m] = Q[n], then we know that Q[n] implies $\Phi \& \Psi$. Using truth-tables, we can show that $\Phi \& \Psi$ implies Φ. If implication is transitive, then it follows that Q[n] implies Φ. And if Q[n] implies Φ, then by definition, line (n) is good, and the rule is sound. To complete the argument, we need to show that implication is in fact transitive. I'll leave this as an exercise for the reader.

Conditional Proof

Here we're given lines (l) and (m), which we assume are good lines, and we need to show that line (n) is a good line.

l	(l)	Φ	A
X	(m)	Ψ	
(X - l)	(n)	$\Phi \rightarrow \Psi$	l,m CP

We know that Q[l] = Φ. There are two different possibilities to consider for the relation between Q[m] and Q[n].

Case 1: line (l) does not occur in dependency numbers for (m). In this case, Q[m] = Q[n].

Case 2: line (l) does occur in dependency numbers for (m). In this case, Q[m] = Q[n] & Φ

Note that in both cases, Q[n] & Φ implies Q[m]. Since, by hypothesis, line (m) is good, then Q[m] implies Ψ. If implication is transitive, then Q[n] & Φ implies Ψ. To finish the proof, we need to show that if Q[n] & Φ implies Ψ, then Q[n] implies $\Phi \rightarrow \Psi$. To show this, we can build a truth table, demonstrating that for any sentences A, B, and C, if A & B implies C, then A implies B $\rightarrow$ C. So the rule is sound. Once again, to complete this argument, we would need to show that implication is transitive.

Universal Elimination

Here we're given line (m), which we assume is good, and we need to show that line (n) is good.

X	(m)	$(v)\Phi(v)$
X	(n)	$\Phi(t)$

Note that Q[m] = Q[n]. Since line (m) is good, we know that Q[m] implies (v)Φ(v). So we know that Q[n] implies (v)Φ(v). To show that line (n) is good—that is, to show that Q[n] implies Φ(t)—we need to show that (v)Φ(v) implies Φ(t) (assuming that we've already shown that implication is transitive). To show this, you'll need to give one of those rigorous informal semantic arguments, like you gave on HW 8, only simpler.

Universal Introduction

We're given line (m), which we assume is good, and we need to show that line (n) is good.

X	(m)	Φ(e)	
X	(n)	(v)Φ(v)	m, UI

Note first that, since we're assuming this is a correctly written proof, the name "e" doesn't occur in Q[m] or in (v)Φ(v). Next, we note that Q[m] = Q[n]. Since, by hypothesis, line (m) is good, we know that Q[m] implies Φ(e). And so Q[n] implies Φ(e). We need to show that Q[n] implies (v)Φ(v).

We'll argue by reductio. Broadly, we'll argue that if Q[n] doesn't imply (v)Φ(v), then by definition, there's an interpretation **I** that makes Q[n] true and (v)Φ(v) false. And if there exists such an interpretation, then it's possible to create a slightly different interpretation **J** on which it also follows that line (m) is bad. Since we're assuming that line (m) is good, interpretation **J** must not be possible, and so **I** must not be possible. So line (n) must be good.

Here are the details:

Suppose that Q[n] does not imply (v)Φ(v). Then there's some interpretation **I** that makes Q[n] true and (v)Φ(v) false. Now, **I** doesn't stipulate any meaning to the name "e", since "e" doesn't occur in line (n). So we'll create a new interpretation **J**, which is just like **I** except that it assigns to "e" whatever object doesn't satisfy (v)Φ(v). We're supposing, remember, that (v)Φ(v) is false on this interpretation—so there must be at least one object that is not Φ. Pick one of them arbitrarily, and assign it to "e".

Now what do we know about **J**? Well, it's just like **I** except that it has this added name "e" referring to something that isn't Φ. Recall that, by hypothesis, **I** makes Q[n] true. And Q[n] = Q[m]. So **I** makes Q[m] true. And we know, since the above proof is written correctly, that "e" does not occur in Q[m]. So **J** makes Q[m] true as well. But **J** also makes Φ(e) false, since we stipulated that "e" refer to something that isn't Φ. So Q[m] doesn't imply Φ(e). So line (m) is bad. But this contradicts our initial assumption that line (m) is good. So, assuming that (m) is good, **J** is not possible. And If **J** is not

possible, then **I** must not be possible. So there is no interpretation that makes Q[n] true and (v)Φ(v) false. So line (n) is good, and our rule is sound.
To complete the project of proving soundness, similar proofs would be needed for all of the rules.

Proving completeness

We want to prove that if some sentence is a consequence of some other sentences—that is, if the argument from the premises to the conclusion is valid—then there is a proof from the premises to the conclusion.

We're going to prove this by proving something stronger.

First, note that whenever you have a valid argument from premises Φ_1, …, Φ_n to conclusion Ψ, the sentence $((\Phi_1 \& \ldots \& \Phi_n) \rightarrow \Psi)$ is a tautology. For example, the argument from premise P to conclusion (P v Q) is valid, and the sentence $P \rightarrow (P \vee Q)$ is a tautology.

Second, note that whenever you have a proof of a tautology of the form $((\Phi_1 \& \ldots \& \Phi_n) \rightarrow \Psi)$, you can get a proof from premises $\Phi_1 \& \ldots \& \Phi_n$ to conclusion Ψ. The proof of the tautology depends on nothing. So once you have that, you can just assume each of Φ_1 to Φ_n, then conjoin them all and use Modus Ponens with the conjunction of premises and the tautology to get Ψ. Ψ will depend on the premises that you assumed.

Putting all of this together: in order to prove completeness, it suffices to show that there's a proof of every tautology. If we can show that, then it follows that there's a proof of every valid argument.

To show that there's a proof of every tautology depending on no premises, we're first going to give a specific strategy for proving each tautology—that is, we're going to show what the proof will look like; and then we're going to prove that this strategy in fact works for every tautology.

For every tautology S, the proof of S is the following sort of reductio:

1	(1)	~S	A	
…	…	…	…	
1	(m)	R	--	where R is a prenex equivalent to ~S
…	…	…	…	from R we derive a finite collection of sentences that imply line (n)
1	(n)	P & ~P		

__	(n+1)	~~S	1,n RAA
__	(n+2)	S	n+1 DN

What do we need to show in order to prove that this strategy works for every tautology? We know that since S is a tautology, ~S is inconsistent. And we know that since ~S is equivalent to R, R is also inconsistent. We need to show the following:

(1) Every sentence has an equivalent in prenex form, and from each sentence, you can derive its prenex equivalent. This gets us from line (1) to line (m).

(2) From every inconsistent sentence in prenex form, you can derive a finite collection of sentences that imply (P & ~P). This gets us from line (m) to line (n).

Notice that, once you show that every sentence has an equivalent in prenex form, you can't prove (1) by simply claiming that all pairs of equivalent sentences are interderivable. Though true, this presupposes completeness—the very thing we're trying to show. Although we won't prove (1) rigorously here, we'll list some of the tools we would need.

On your handout on prenex form, there are ten different forms of sentences containing one quantifier, together with their prenex equivalents. To prove (1), we could start by deriving each prenex equivalent from its original non-prenex sentence. Then we need to generalize this result to sentences containing different predicate letters and variables, and sentences containing multiple quantifiers. To do this, we would need the following three laws:

Law of Alphabetic Variance: roughly, if you change the letters of the variables of some sentence, you get an equivalent sentence. So (x)Fx is equivalent to (y)Fy.

Law of Interchange: if you replace a propositional function that occurs in some sentence S with a quasi-equivalent propositional function, the resulting sentence S' is equivalent to S. (Whereas equivalence is a property of pairs of sentences, quasi-equivalence is a property of pairs of propositional functions. Two propositional functions are quasi-equivalent if and only if the sentences that result by attaching the same quantifier to both propositional functions are equivalent. For example, the propositional functions (Fx → Gx) and (~Fx v Gx) are quasi-equivalent; the sentences (x)(Fx → Gx) and (x)(~Fx v Gx) are equivalent.)

Law of Substitution: if you have a valid argument, and you systematically replace any predicate letter with a simple or complex predicate of the same arity (that is, the same number of argument spaces—Fx and Gy have the same arity, Fxy and

> Gxy have the same arity, etc.), the resulting argument will be valid. For example, if you replace all occurrences of 'Fx' in some valid argument with '(x)(Fx →Gy)', the resulting argument will be valid.

That goes some way toward showing how you could prove (1). But the heart of the completeness proof lies in proving (2). Here too we're not going to go so far as to prove (2). Instead, we'll show what tools we'll need.

The primary tool is an algorithm, similar to Algorithm B, which will generate a possibly infinite list of quantifier-free sentences. The algorithm works as follows:

> You're given an infinite stock of variables a_1, a_2, etc.
>
> At step (1), you start by writing your prenex sentence down.
>
> Next, you instantiate the outermost quantifier using a_1.
>
> After any step (n) in the algorithm, you get to step (n+1) by (a) writing down instances of each existential sentence in step (n) using the first variable not already used; and (b) writing down the instances of each universal sentence obtained up through step (n) using every variable not already used at any step through step (n).

For example, suppose you start with the sentence $(x)(y)(Ez)\Phi(x,y,z)$. The list of sentences would look as follows:

Step 1 $(x)(y)(Ez)\Phi(x,y,z)$
Step 2 $(y)(Ez)\Phi(a_1,y,z)$
Step 3 $(Ez)\Phi(a_1,a_1,z)$
Step 4 $\Phi(a_1,a_1,a_2)$
Step 5 $(y)(Ez)\Phi(a_2,y,z)$
$(Ez)\Phi(a_1,a_2,z)$
Step 6 $(Ez)\Phi(a_2,a_2,z)$
$\Phi(a_1,a_2,a_3)$

…and so on.

Now, there are a couple of facts that we would need to prove in order to rigorously prove completeness:

> If the initial quantified sentence is inconsistent, then the collection of quantifier-free sentences that we end up with is inconsistent.
>
> Further, it also turns out that if some infinite set of sentences is truth-functionally inconsistent, then some finite conjunction of members of that set is truth-

functionally inconsistent. (This very important result is known as the Compactness Theorem.)

Given these facts, it follows that if R is some inconsistent sentence in prenex form, then the our little algorithm will deliver some finite set of quantifier-free sentences that are inconsistent.

And now we're almost home. In order to show that we can prove (P & ~P) from R, we need two more points.

First, we need to assume that the propositional calculus is complete. Specifically, we're going to assume that if you have a conjunction of inconsistent quantifier free sentences, then there's a proof from those sentences to (P & ~P).

Second, we need to note that our algorithm can be replicated within an actual proof as follows: all of the instantiations that take place in the algorithm are either uses of UE or they are assumptions of instances of existential sentences. Once we prove (P & ~P), which doesn't contain any variables, we can go through a series of steps of EE on (P & ~P), for each instance of an existential sentence.

So from any inconsistent prenex sentence R, we can derive (P & ~P). Once we've got that, we've proven (2).

And once we've proven (1) and (2), we've succeeded in showing that there is a proof for every tautology. And so there is a proof of every valid argument.